Intent Based Utterance Segmentation for Multi Intent NLU

A THESIS
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA
BY

Arun Sethupat Radhakrishna

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE

Dr. Joseph Konstan and Dr. Serguei Pakhomov

August, 2019

Acknowledgements

I would like to extend my sincere thanks, gratitude and appreciation to all those who have guided and assisted me in the completion of this degree.

Firstly I would like to thank my advisor Prof. Joseph Konstan for all the guidance and support that he has provided throughout my Masters Degree. His timely advice has helped me to push myself to do more.

Secondly, I wish to thank my co-advisor Prof. Serguei Pakhomov for all the feedback and guidance provided to me throughout the course of my research. His guidance during my research helped me overcome many obstacles.

In addition, I would also like to express sincere gratitude to Prof. Maria Gini for being on my committee and providing valuable feedback.

Finally, I would like to thank my parents, Mr. Radhakrishna Sethupat and Mrs. Lakshmi Sethupat, my best friend, Ms. Krishna Pranathi Mokshagundam, my brother Tarun Sethupat and my mentor Kuldeep Singh for their constant unwavering support throughout my studies.

Dedication

To my parents.

Abstract

Natural Language Understanding(NLU) is a process of converting the user utterance to a dialog-act after identifying domain, intent and slots from the utterance. User utterances can either contain a single intent or could express multiple intents. Building an NLU module for multi-intent utterances is a huge challenge as traditional state-of-the-art NLU modules do not differentiate between single and multi intent utterances thereby converting them to a single semantic frame which results in reduced performance. In this paper, we introduce a intent based utterance segmenter to split user utterances if each segmented clause corresponds to a different intent. Our experiments evaluate the performance of the utterance segmenter not only on the utterances from movie-ticket booking domain and restaurant reservation domain used for training but also on a new taxi ordering domain. We show that the total number of utterances that are parsed by a utterance segmenter enabled NLU surpass the utterances parsed by traditional NLU.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Need for a novel NLU approach

Natural Language Understanding is the first processing step in any text based dialogue system. These components are responsible for converting the user utterance into a semantic form called dialogue acts. A dialogue act consists of user intent and slot values(information required to complete the task). Current state-of-the-art NLU systems specialize in converting a user utterance into one single dialogue act. However in reality, the user utterance could have multiple intents with associated slots. Ability to identify all the dialogue acts from the user utterance would mean that the performance of NLU component is improved which in turn would significantly improve the performance of the dialogue system.

1.2 Scope

Dialog systems are of three types [1] 1. Question Answering: A system which can retrieve a definite answer to a user query. 2. Task Oriented: A system which can perform a specific task based on user request(Eg: Movie ticket booking) 3. Social Bots: A system which keeps the user engaged with relevant responses. In this thesis we focus on Task Oriented Dialogue Systems. Hence any reference to a dialogue system would mean Task Oriented Dialogue System. Task Oriented Dialog systems rely on slot filling and are usually made up of the following components.

- Natural Language Understanding: This module takes the users raw utterances as input and converts them to dialogue acts.
- Dialog State Tracker(DST): This module is responsible for keeping track of the current state of the conversation. DST takes the dialogue act generated by the NLU as input.
- Dialog Policy: This module, the policy, relies on the internal state provided by DST to select an action. Note that an action can be a response to the user, or some operation on backend databases
- Natural Language Generation: If the policy chooses to respond to the user, this module will convert this action, often a dialogue act, into a natural language form.

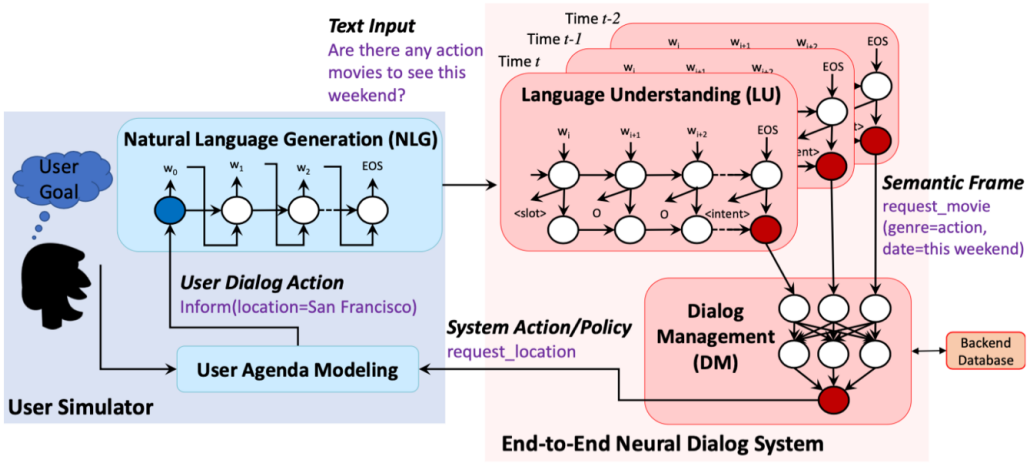

Figure 1.1: Components in a dialogue system

We restrict our work to Natural Language Understanding Component in Task Oriented Dialog Systems. We further focus on the intent recognition aspect of NLU. Li [2] notes that the intent prediction by NLU is the most important task for the success of the dialog system. Improving Intent recognition would significantly improve the overall functioning of the Dialog system. However, we also make sure that the experiments we conduct will in no way inhibit the working of the slot filling component of NLU.

1.3 Summary of Contributions

In this thesis, we propose an algorithm which segments utterances in a way such that each utterance segment would express a definitive user intent. We propose a method which a) works with small user utterances, b) is scalable and c)would potentially work with utterances from different domains. We show that having a intent based utterance segmentation as a pre processing step before being parsed by NLU significantly improves the percentage of utterances that can be parsed by NLU.

1.4 Thesis Organization

Chapter 1 introduces the need and scope of the methods proposed. Chapter 2 explains the motivation behind choosing this problem with an example. In Chapter 3 we detail the problem statement that we wish to address with this thesis. Chapter 4 talks about the basic concepts necessary to understand the work and also details the available literature for each concept. In Chapter 5, we explain how the datasets were selected and curated. Chapter 6 talks in detail about the model used and how it performs on various datasets and evaluation settings. Finally we conclude by highlighting the contributions in Chapter 7.

Chapter 2

Motivating Example

During an interaction with a movie-ticket booking dialogue system, consider a case where user responds with the following utterance. *The Zipcode is 48126.. was also looking for 4 tickets for batman vs superman movie.* Traditional NLU systems would completely ignore the first part of the sentence and would understand that the user is requesting for 4 tickets for Batman vs Superman movie as shown in 2.1

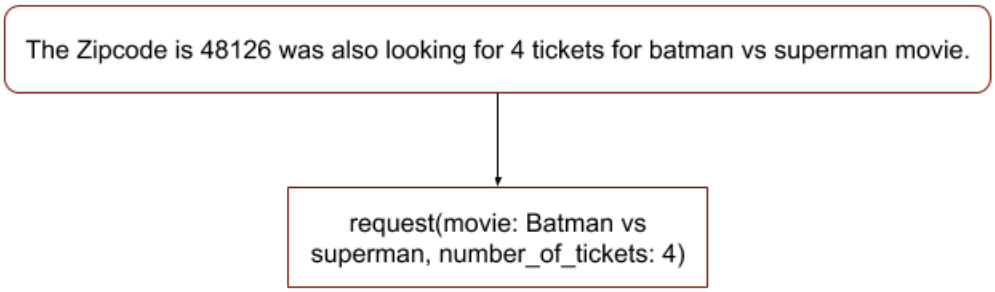

Figure 2.1: Traditional NLU parser

Traditional systems ignore the first part because they are caliberated to identify only one intent from the utterance. Since there are two intents present in the utterance, only one among them would be processed and the other one would be ignored. However, if we understood the other piece of information the user gave(information about the zipcode), it would have enabled the system to shortlist the movie hall locations based

on the zipcode information.

Hence, if the user utterance is segmented before being parsed by the system, such that each segment in the utterance would correspond to a different intent, then the system would parse both pieces of information separately as shown in 2.2

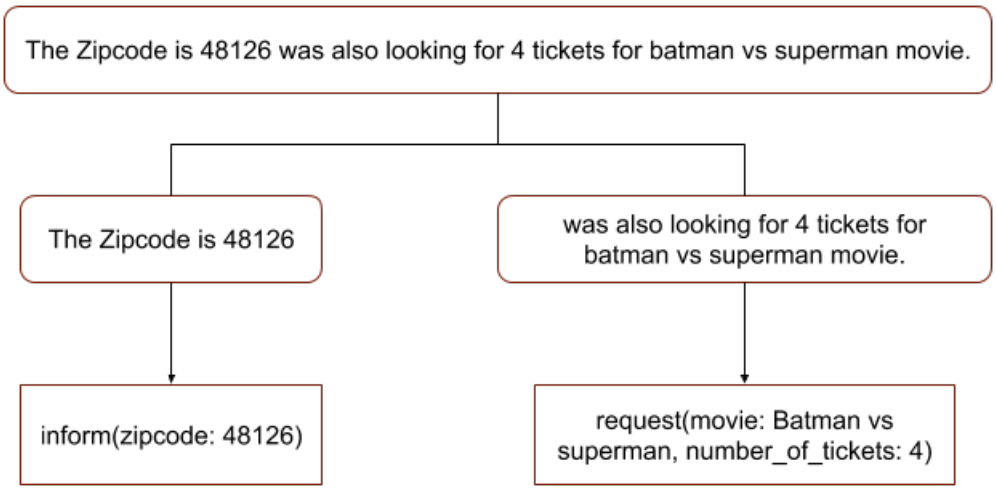

Figure 2.2: Segmenter enabled NLU parser

The user input *The Zipcode is 48126. was also looking for 4 tickets for batman vs superman movie.* is segmented into two parts *The Zipcode is 48126* and *was also looking for 4 tickets for batman vs superman movie.*. Both phrases are parsed by NLU separately and both pieces of information can be used by the chatbot.

Chapter 3

Basic Concepts and Literature Review

3.1 Machine Learning Concepts

3.1.1 Artificial Neural Networks

Artifical Neural Networks are interconnected assemblies of simple processing units called nodes. The processing ability of the network is stored in inter unit connection strengths, called weights, obtained by a process of adaption called learning, from a set of training patterns. In other words, ANNs are computational systems that use a network of functions to understand and translate a data input of one form into a desired output [3]. For more detailed understanding of these networks, refer Gurney et. al [3].

3.1.2 Language Models

Models that assign probabilities to a sequence of words are called Language models [4]. The simplest model that assign probabilities to a sequence of words or a sentence is called n-gram model. An n-gram is a sequence of N words: a 2-gram (or bigram) is a two-word sequence of words like please turn, turn your, or your homework, and a 3-gram (or trigram) is a three-word sequence of words like please turn your, or turn your homework. These models are generally used to estimate the probability of the last word of the n-gram given the previous n-1 words.

We estimate the parameters of n-gram models using the training data. If W is a sentence containing series of words $w_1, w_2, ..., w_n$ and $C(w_1)$ is number of times word w_1 occurs in the training set, then we estimate the parameters of an n-gram model as follows [5].

$$Pr(w_n|w_1^{n-1}) = \frac{C(w_1, w_n)}{\Sigma_w C(w_1, w)} \tag{3.1}$$

where

$$P(w_1^n) = \Pi_{k=1}^n P(w_k|w_{k-1}) \tag{3.2}$$

This method is called sequential maximum likelihood estimation [5]. These methods are used for applications such as natural language prediction [5], lexical disambiguation [6], machine translation [7] and word embeddings. [8] [9].

3.1.3 Activation Functions

At the heart of every deep network lies a linear transformation followed by an activation function $f()$. The activation function plays a major role in the success of training deep neural networks. Activation functions takes an input and defines a specific output. Non linear functions allows the neural networks to learn with smaller number of nodes. Currently, the most successful and widely used activation function is Rectified Linear Unit (ReLU) [10]. We discuss Sigmoid, ReLU and Leaky ReLU variants here as they are used in our model architecture. [11]

Sigmoid

Sigmoid activation function is closed, differentiable, bound function which has a positive output value for all the input values. This is usually used in binary classification problems.

$$f(x) = \frac{1}{1 + e^{-x}} \tag{3.3}$$

ReLU

Rectified Linear Unit often called ReLU [12] is a piecewise linear function which outputs the input directly if the input is positive else will output zero. This has been shown to

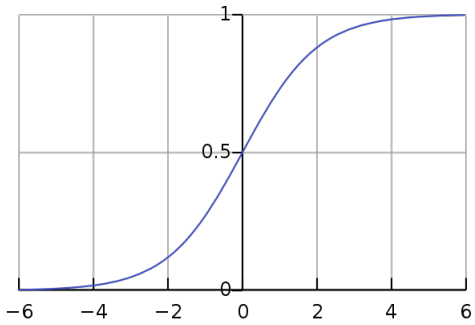

Figure 3.1: Sigmoid Activation Function

be the most effective activation function because it of the performance improvement it provides and the ease of training.

$$f(x) = \begin{cases} x & x > 0 \\ 0 & x \leq 0 \end{cases}$$

Leaky ReLU

Leaky ReLU [13] is an activation function which also outputs the input value if the input is positive but also allows a small gradient to propagate if the input is negative.

$$f(x) = \begin{cases} x & x > 0 \\ 0.01x & x \leq 0 \end{cases}$$

ReLUs have the problem of creating a lot of dead cells if the input value is negative or if the weights are not properly initialized. This would mean that the network doesn't train at all. Leaky ReLUs avoid that problem by propagating a small negative gradient and there by allowing the training process to happen without the fear of improper initialization.

3.1.4 Optimization Algorithms

An optimization algorithm is a procedure which is executed iteratively by comparing various solutions until an optimum or a satisfactory solution is found. In the context of

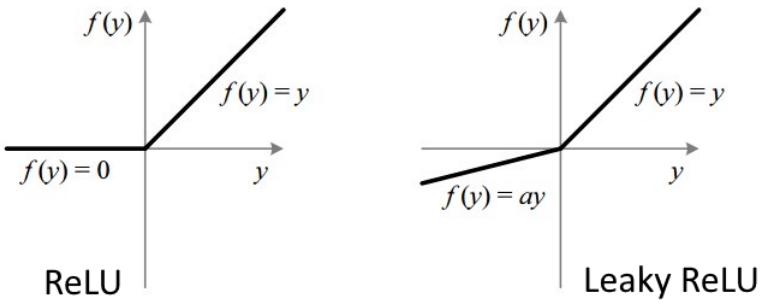

Figure 3.2: ReLU and Leaky ReLU Activation Function

deep learning we use optimization functions to optimize the cost function and thereby minimize the loss function. Stochastic Gradient Descent [14] has been the most popular optimization algorithm however in many deep networks it has been found to be ineffective since it lacks the ability to optimize if there are saddle points. [15].

Adam Optimizer

Adaptive Moment Estimation(Adam) [16] is an method that computes adaptive learning rates for each parameter. In addition to storing an exponentially decaying average of past squared gradients v_t,Adam also keeps an exponentially decaying average of past gradients m_t , similar to momentum. Whereas momentum can be seen as a ball running down a slope, Adam behaves like a heavy ball with friction, which thus prefers flat minima in the error surface [17].

For further understanding of performance of Adam Optimizer and similar such optimizers, refer [15].

3.2 Word Embeddings

A word embedding is a learned representation for text where words that have the same meaning have a similar representation. Word embedding techniques essentially represent individual words as real-valued vectors. These techniques improve the ability of neural networks to learn from textual data. These word embeddings are usually pre-trained with an aim to capture certain word attributes.

For example, GloVe [18] uses an unsupervised learning algorithm trained on aggregated global word-word co-occurrence statistics from a corpus to showcase interesting linear substructures of the word vector space.

Fasttext [8] is trained using character n-gram models so that the trained models can work with rare-words and words which were not available during the training phase.

Word2Vec [19] and Global Context and Multiple Word Prototypes based embeddings by Huang et. al. [20] are other such embedding techniques.

3.3 Natural Language Understanding(NLU)

Natural Language Understanding(NLU) is a subtopic of NLP that deals with Machine Comprehension. As noted by Li [2] an NLU component, in general, is expected to

- identify the domain of the user query,
- identify the domain specific intents
- fill in a set of slots to form a semantic frame

NLU is a pre-processing step for later modules in the dialogue system, whose quality has a significant impact on the systems overall quality. [2]

W	find	action	movies	this	weekend
	↓	↓	↓	↓	↓
S	O	B-genre	O	B-date	I-date
I	find_movie				

Figure 3.3: IOB format sentence labelling

A popular IOB(In-Out-Begin) Format is used to represent the slot tags. Most of the state-of-the-art NLU systems [21], [22], [23], [24] specialize in converting an input utterance into a single semantic frame. However in chat based or voice based dialogue system applications, the user utterances could have multiple clauses such that each clause expresses a different intent and also has associated slot-tags.

3.4 Multi Intent Understanding

In this section we discuss different methods employed to identify the presence of multiple intents from a single utterance.

New labels

Ruhi Sarikaya et.al. [25] talk about a method in which multiple primary intents labels are combined to form new intent labels. i.e if request and inform are two intent labels present in the domain, request+inform, request+request and inform+inform would be other intent labels that are added to the domain. The approach works well to some extent but has the following drawbacks.

- The system does not scale well, we would need to add many such new labels with the increase in the number of primary intents.
- Enough training data is required for each label separately, we do not take advantage of the information we already have about the primary labels.
- Even if we successfully identify the different intents, slot filling is still a challenge.
- The work is restricted to only 2 intent phrases.

Multi label classification

Another approach suggested is to perform a multi-label classification. This method would identify the multiple labels present in the single multi-utterance phrase. Biggest advantage of such an approach is that it successfully leverages the information we have about the primitive labels. However, this method too comes with a certain set of drawbacks.

- Since this is a multi-label classification problem, we can identify the presence of multiple different intent labels. However we cannot identify the number of times each intent is present. i.e. request+request cannot be identified.
- Slot filling cannot be performed.

- Although this work can be extended to multiple intent sentences, the author restricts his analysis to only 2 intent phrases.

Splitting Sentences for NLU

In order to overcome this problem, Kim et.al. [26] suggest splitting sentences before parsing by NLU. He talks about identifying intents in two stages, in the first stage he segregates utterances with conjunctions to split and in the second stage he identifies the presence of intents in each utterance. The advantage of this method is that we would not need any additional training data for sentences with multiple intents and splitting sentences help in slot filling. However this method comes with the following drawbacks.

- Splitting of sentences is completely rule based and relies on the presence of punctuation marks and conjunctions.
- Work is restricted to the space of 1 and 2 intent phrases.

Observation

Out of all the multi intent identification methods discussed in this section, splitting utterance for NLU is the only approach which is promising. This is because splitting uttreances would help us not only in identifying the multiple intents(even if the intents are duplicate) but would also not inhibit the slot filling process. However even the sentence splitting technique used here is completely rule based and is limited to at most 2 intents in an utterance.

In the next section we would survey the different segmentation approaches available and identify their drawbacks.

3.5 Sentence Segmentation

Sentence Segmentation also commonly referred to as Sentence Boundary Disambiguation and Sentence Boundary Detection is the process of identifying boundaries between a group of words in a text document such that each group inside a boundary corresponds to a meaningful linguistic unit. This is a useful first step in developing most

NLP task applications such as morphological analyzers, parts of speech taggers, text parsers, speech to text converters and information retrieval systems. [27]

Apache OpenNLP

Apache OpenNLP [28] takes plain text as input and implements a rule based sentence segmenter originally inspired by Stanford English Penn Treebank tokenizer but with several modifications and improvements. This takes advantage the presence of punctuation marks and word cases to identify the boundaries.

NLTK

Natural Language Toolkit [29] commonly referred to as NLTK provides a sentence segmentation tool which takes raw textual input and identifies multiple clauses if there are any. This method also relies on presence of punctuation marks and word casings. NLTK also provides Punkt, a sentence segmentation tool which allows us to train the module with custom data.

CNN-BiLSTM based segmenter

Knoll et. al. [30] propose a solution that relies on deep CNN-BiLSTM network architecture. This method works even in cases where there is no presence of punctuation marks. This architecture works well in documents containing multiple sentences(paragraphs). The author uses two models trained using two datasets(MIMIC III and Fairview datasets).

Other methods

[31] presents an approach to identifying sentence boundaries. They propose finite state models that extract sentence boundary information statistically from text and audio sources using pause detection models. [32] also proposes a method for sentence boundary disambiguation in speech using HMMs and pause detection models.

[33], [34] and [35] propose sentence segmentation methods which are specific to domains such as finance, legal and news.

Various other authors [36], [37], [38] discuss methods to split long or ill-formed sentences. However, all these methods rely on the presence of punctuation marks given in the utterance or the presence of sufficient context to split the sentences.

Table 3.1: Sentence Segmentation methods

Method	No Punctuation	No Capitalization	Short Text
OpenNLP	No	No	Yes
NLTK	No	No	Yes
Mimic - CNN-BiLSTM	Yes	Yes	Not ideal
FV - CNN-BiLSTM	Yes	Yes	Not ideal

Based on availability of resources and APIs along with the relevance of the methods described we compare our approach with OpenNLP, NLTK, CNN-BiLSTM MIMIC and CNN-BiLSTM FairView datasets.

Chapter 4

Problem Statement

To develop a novel method which can improve the performance of Natural Language Understanding modules for utterances containing multiple intents by augmenting the NLU component with an intent based utterance segmentation module which can

- Use simple utterances for training : Since this task is data intensive and it is difficult to obtain utterances containing multiple intents, it is important to leverage the simple user utterances to predict intent based clause boundaries.
- Help slot-filling: NLU task performs both intent prediction and slot filling. A model which improves the performance of intent prediction at the expense of slot filling performance is not desirable.
- Identify more than 2 intents: A approach which can be efficiently scale for complex utterances containing any number of intents is desirable.
- Identify duplicate intents: A user utterance can have multiple sentences which correspond to same intent. A system which identifies the presence of both the intents and yet does not inhibit slot-filling performance is desirable.
- Is scalable: A method which doesn't improve the execution overhead for complex utterances.
- Is cross-domain: A method which works not only for the domain it is trained for but works for other domains also is desirable. This is critical because it is difficult

to collect and annotate data for each domain.

- Work without the presence of punctuation marks: User typed utterances are usually not grammatically correct and hence a method which doesn't use punctuation marks as clues for segmenting is desirable.

Chapter 5

Corpus Creation

For this experiment we use the conversational data released by Microsoft as a part of Microsoft Dialogue Challenge at SLT2018 [39]. This dataset is a collection of utterances of users and the chatbot along with corresponding intent-slot pairs as labels in a task oriented dialog system setting. The data is from movie-ticket booking, restaurant reservation and taxi booking domains. Each response in the dataset could have multiple intent-slot pairs as labels. Table 6.8 lists the count of multi-intent and single-intent responses in each domain.

Table 5.1: Domain wise utterance count

	Movie-ticket booking	Restaurant reservation	Taxi Ordering
Single intent sentence	14534	22357	18572
Multi intent sentence	5540	3165	3810
Total sentences	20074	25522	22382

5.1 Data Cleaning and annotation

The objective of this step is to identify the sentences with multiple intents and then identify the word token after which the sentence should be split. For example: Consider the sentence *The Zipcode is 48126. was also looking for 4 tickets for batman vs superman movie.* We used the following rule based technique to identify the word token after which the sentence should be split.

- Presence of periods(.), question marks(?), exclamation mark(!) and comma(,) in the respective order of precedence.
- Presence of conjunctions such as 'and', 'or', 'therefore', etc.

Once the sentence splits have been recorded by all the sentences, we compare the number of phrases present in each sentence with the number of intent labels. We discard all the sentences where there is a mismatch. This step is done to ensure that the only sentences that are rightly split are considered for further evaluation.

5.2 Training and evaluation datasets

We use 80% percent of the data from movie-ticket booking and reservation domain for training and the remaining 20% data for testing the model. We also use the taxi ordering dataset for testing the models on new domain that wasn't introduced during model training. All the testing datasets have been stratified.

Table 5.2: Training dataset utterance counts

	Movie-ticket booking	Restaurant reservation	Taxi Ordering
Single intent utterances	11646	17870	0
Multi intent utterances	4413	2547	0
Total sentences	16059	20417	0

Table 5.2 shows the size of data used for training from each domain. Note that the size of data used for training from Taxi domain is 0. These sentence datasets are further modified to meet the experiment requirements. The modifications will be discussed in the experiment section.

Chapter 6

Experiment

The objective of this experiment is to identify the location of the split in the utterances if there are multiple clauses such that each clause corresponds to different intent labels. In order to do this, we first modify the input data such that it can be used to train a neural network.

6.1 Data modification

We use the training and testing datasets created in Chapter 5 and then modify them in the following manner. These steps are explained in 6.1

- Step 1: Read the input sentence.
- Step 2: Remove all the punctuation marks and convert the upper case letters to lower case letters.
- Step 3: Append the beginning of the sentence with two *bos* tags.
- Step 4: Create a new data point for every word with 2 words preceding the current word and 1 word following the current word as the context for the word. And the label for the data point being 1 if the sentence splits after the word or 0 if there is no split in the sentence after this word.
- Step 5: Transform the words in the datapoints to word vectors using pretrained gloVe model. We use 50 dimensional gloVe word vectors.

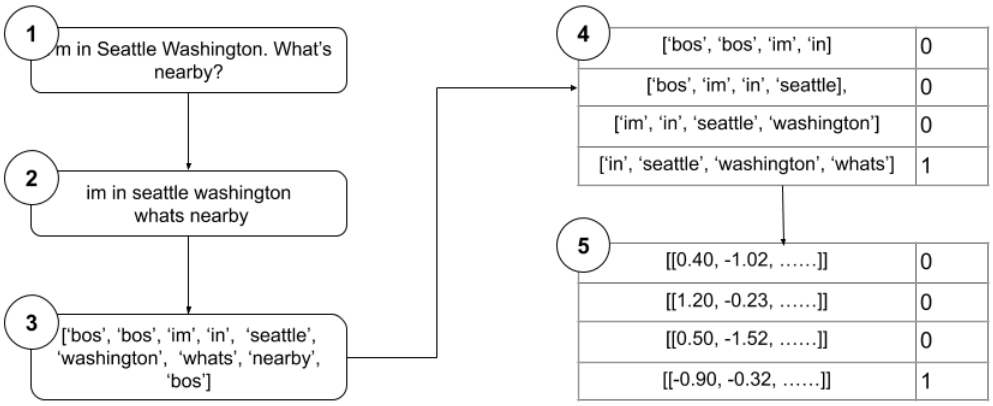

Figure 6.1: Data modification steps

6.2 Model Architecture

We build an artificial neural network with 2 hidden layers with 100 nodes in each layer. Each data point used for training has the dimension of 4*50. We flatten this data point such that the dimension of the input data point is 200*1 and hence the input layer now consists of 200 nodes. We use Leaky Rectified Linear Units(ReLUs) as activation functions for each node in every layer except the last layer. The final layer consists of 2 nodes such that one node effectively calculates the probability of spliting the utterance for this point where as the other node calculates the probability of not splitting. Since this is a binary classification problem, the last layer is activated by sigmoid activation function.

The model was built using Keras and tensorflow modules in Python 3.6. It was trained using the training dataset created using techniques mentioned in 5 and modified using techniques mentioned in 5.1. We set the number of epochs to 10. The optimizer used was Adam.

6.3 Metrics

The performance of a classifier is evaluated using following metrics.

1. Precision: For a positive class, precision is the ratio of correctly predicted positive

observations to the total predicted positive observations.

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives} \tag{6.1}$$

2. Recall: For a positive class, Recall is the ratio of correctly predicted positive observations to the all observations in actual class.

$$Recall = \frac{TruePositives}{TruePositives + FalseNegatives} \tag{6.2}$$

3. F1 Score: F1 Score is the weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account.

$$F1Score = \frac{2 * Precision * Recall}{Precision + Recall} \tag{6.3}$$

4. Accuracy: Accuracy is the most intuitive performance measure and it is simply a ratio of correctly predicted observations to the total observations.

$$Accuracy = \frac{TotalCorrectPredictions}{TotalObservations} \tag{6.4}$$

5. Weighted Metrics Weighted Metrics are the average metrics of all classes weighted by number of samples of each class. These metrics provides a realistic view of the performance of the classifiers which have class imbalances. Suppose there are n classes and w_i be the number of samples from class i then

$$WeightedPrecision = \frac{w_i * Precision_i}{\Sigma_{i=0}^{n} w_i * Precision_i} \tag{6.5}$$

$$WeightedRecall = \frac{w_i * Recall_i}{\Sigma_{i=0}^{n} w_i * Recall_i} \tag{6.6}$$

$$WeightedF1Score = \frac{w_i * F1_i}{\Sigma_{i=0}^{n} w_i * F1_i} \tag{6.7}$$

6.4 Token Level Evaluation

Objective

The aim of this evaluation is to evaluate the classification performance of the model built with specifications mentioned in 5.2. The total size of the dataset used for training is 339248. As mentioned previously, these tokens are only collected from movie-ticket booking and restaurant reservation domains. Tokens from taxi-ordering have not been used.

Table 6.1 shows the total number of tokens present in testing datasets. There are 6720 randomly sampled tokens from movie-ticket booking domain, 3542 tokens from restaurant reservation domain and 22304 tokens from taxi-ordering domain. In all these domains, 50% of the tokens have splits after the given token and 50% words do not split after the token.

Table 6.1: Token level experiment test dataset counts

	Movie-ticket booking	Restaurant reservation	Taxi Ordering
No split tokens	3360	1771	11152
Split Tokens	3360	1771	11152
Total tokens	6720	3542	22304

Classification Performance

On all the datasets, we emphasize on two metrics 0-class(No Split) recall and 1-class(Split) case precision. This is because we would be introducing more error to the NLU module if we wrongly split the sentence after the given token. However, if we miss the split after the given token, that would not affect the NLU performance as sentences with splits were anyway not processed by the traditional NLUs. Also, since a typical sentence contains more no split tokens than split tokens we would want our no split recall to be high.

Table 6.5 shows that in the movie-ticket booking domain the average precision is 0.95, average recall is 0.95 and the average F1 Score also as 0.95. We should note that

both the no-split recall and split precision is 0.98.

Table 6.6 shows that in the restaurant reservation domain the average precision is 0.94, average recall is 0.93 and the average F1 Score also as 0.93. We should note that both the no-split recall and split precision is 0.98.

Table 6.2: Evaluation of Movie-Ticket Booking test dataset

Movie Ticket Booking	Precision	Recall	F1 Score
0 Class(No Split)	0.92	0. 98	0.95
1 Class(Split)	0.98	0.92	0.95
Weighted Average	0.95	0.95	0.95

Table 6.3: Evaluation of Restaurant Reservation test dataset

Restaurant Reservation	Precision	Recall	F1 Score
0 Class(No Split)	0.89	0. 98	0.93
1 Class(Split)	0.98	0.88	0.93
Weighted Average	0.94	0.93	0.93

In case of taxi ordering dataset which has not been used during the training phase at all, has recorded an average precision of 0.96, average recall of 0.97 and average F1 Score of 0.97. The no-split recall is found to be 0.98 and split-precision is found to be 0.97.

Table 6.4: Evaluation of Taxi Ordering dataset

Taxi Ordering	Precision	Recall	F1 Score
0 Class(No Split)	0.89	0. 97	0.93
1 Class(Split)	0.97	0.88	0.92
Weighted Average	0.93	0.92	0.93

This shows that the performance of the model on domains which are visible during the training phase and the performance of the model on domains hidden during the

training phase are comparable.

6.5 Utterance level evaluation

Objective

The objective of this evaluation is to report the classification performance on all the tokens from the sentences in the testing set. This would mean that the classification performance will be calculated on unbalanced number of tokens. On an average each utterance has around 10 tokens without a split and one token where there is a split. However, the total number of utterances in each test dataset have equal number of sentences with splits and sentences with no-splits.

Classification Performance

We can see that in the real setting i.e. a case where we have more tokens with no-split labels that tokens with split labels, there is an improvement in the performance of the model.

In case of movie-ticket booking domain the weighted average precision is 0.98, weighted average recall is 0.97 and weighted average F1-Score is 0.98. The no-split recall ans split precision are still considerably high.

In case of restaurant reservation domain the weighted average precision is 0.97, weighted average recall is 0.97 and weighted average F1-Score is 0.97. The no-split recall is 0.98 and split precision is 0.85. In both the domains the drop in split precision can be attributed to increase in the no-split tokens.

Table 6.5: Evaluation of Movie-Ticket Booking test dataset

Movie Ticket Booking	Precision	Recall	F1 Score
0 Class(No Split)	0.99	0. 98	0.99
1 Class(Split)	0.90	0.92	0.91
Weighted Average	0.98	0.97	0.98

Table 6.6: Evaluation of Restaurant Reservation test dataset

Restaurant Reservation	Precision	Recall	F1 Score
0 Class(No Split)	0.99	0. 98	0.98
1 Class(Split)	0.85	0.88	0.87
Weighted Average	0.97	0.97	0.97

Where as in the case in taxi ordering domain, the weighted average precision is 0.97, weighted average recall is 0.96 and weighted average F1-Score is 0.96. The no-split recall is 0.97 and split precision is 0.79. The performance of the taxi data set(hidden while training) is comparable to the movie-ticket booking and restaurant reservation domain datasets.

Table 6.7: Evaluation of Taxi Ordering dataset

Taxi Ordering	Precision	Recall	F1 Score
0 Class(No Split)	0.98	0. 97	0.98
1 Class(Split)	0.79	0.88	0.83
Weighted Average	0.96	0.96	0.96

Utterance level Accuracies

Here we run the classifier on all the tokens in an utterance and we count the utterance to be accurately split if all the tokens are correctly predicted by the classifier. Even if there is a single token in an utterance that is incorrectly split that utterance would be counted as incorrectly parsed utterance.

We compare our utterance level performance with MIMIC-CNN-BiLSTM Sentence Segmenter, NLTK, OpenNLP and Fairview- CNN-BiLSTM Sentence Segmenter.

Table 6.8: Utterance level Accuracies

Algorithm	Movie-ticket booking	Restaurant reservation	Taxi Ordering
MIMIC	0.40	0.39	0.40
NLTK	0.50	0.50	0.50
OpenNLP	0.50	0.50	0.50
Fairview	0.51	0.55	0.51
N-gram based splitter	0.78	0.67	0.62

We note that NLTK and OpenNLP sentence segmentation modules can process only 50% of the sentences in the datasets. This is because these methods rely completely on punctuations and word cases. Since 50% of our sentences do not have punctuations, these modules process them correctly. Whereas, MIMIC-CNN-BiLSTM and Fairview-CNN-BiLSTM methods can process sentences without punctuations and word cases. MIMIC-CNN-BiLSTM records performances of 0.40 and 0.39 on movie-ticket booking and restaurant reservation domains whereas Fairview-CNN-BiLSTM records performance of 0.51 and 0.55 on movie-ticket booking and restaurant reservation domains. Our N-gram based segmenter method outperforms all these methods by recording performances of 0.78 and 0.67 on both these domains respectively.

However, this is an unfair comparison as movie-ticket booking and restaurant reservation domains data was used to train the N-gram based segmenter. On the taxi ordering dataset we see that MIMIC-CNN-BiLSTM and Fairview-CNN-BiLSTM record performances of 0.40 and 0.51 respectively and OpenNLP and NLTK record performance of 0.50. Where as N-gram based segmenter performs significantly better than all these methods by recording a performance of 0.62.

Chapter 7

Conclusion and Discussion

7.1 Discussion

The experiments in Chapter 6 show that intent based utterance segmenter segments multi-intent utterances effectively in comparison with other segmentation tools. This module when used to augment the traditional NLU component, would increase the overall performance of the system.

Table 7.1 shows that out of 2254 utterances in the movie-ticket booking domain, the total number of utterances parsed by traditional NLU are 1127 whereas 1747 utterances were parsed by utterance segmenter augmented NLU. Similarly of 1236 utterances from the restaurant reservation domain 668 utterances were parsed by the traditional NLU whereas 800 utterances were parsed by utterance segmenter enabled NLU. Finally our experiments on the utterances from taxi ordering domain show that 7618 utterances traditional NLUs parse 3809 while intent based utterance segmenter enabled NLU parse 4690 utterances.

Table 7.1: Total number of utterances parsed by NLU

	Movie-ticket booking	Restaurant Reservation	Taxi Ordering
Traditional NLU	1127	668	3809
Utterance Segmenter enabled NLU	1747	800	4690
Total Utterances	2254	1236	7618

Intent based utterance segmentation augmented NLU comes with the following advantages

- Uses single intent utterances for training and hence becomes easier to train. This is because it is very common to find single utterance sentences.
- Helps slot filling: Unlike other methods which identify multiple intents, this method makes it easy for the NLU component to identify the associated slots since each clause is treated separately which avoids ambiguity for the slot values.
- This system can easily identify the presence of more than 2 intents in a given sentence. This is another major contribution since all methods demonstrate the performance on atmost 2 intents.
- Since the utterance is separated and parsed separately by the NLU, the presence of duplicate intents would avoid not only the additional overhead to the system but would also not inhibit slot filling.
- This method works across different domains. We showed that the segmenter trained on utterances from movie-ticket booking and restaurant reservation domain worked with utterances from taxi ordering domain too.
- The biggest advantage of the intent based utterance segmenter is that it works without the presence of punctuation marks. Most available segmentation techniques fail when the utterances are not punctuated.

7.2 Conclusion and Future Work

Although augmenting the NLU component with intent based utterance segmentation module improves with the overall functioning of the NLU system, it still has few considerable drawbacks.

- The system adds additional processing overhead to the dialog system. Dialog systems are near real time systems and hence their response time is of significance. Adding additional components would increase the processing time of the system. Hence steps are needed to increase the efficiency of the segmentation module.
- Although augmenting the NLU component with intent based utterance segmentation module works with cases when each utterance segment corresponds to unique intent and hence belongs to unique dialog-act. However, in reality there could be utterance segments which could express or add information to the same intent. However, in practice, we found that such utterances are limited and hence was not the focus of the discussion. Nevertheless, a system which could identify such utterances would be ideal.
- Since the utterance segmentation is a pre-processing step before being parsed by NLU, the errors from the utterance segmentation stage impact the efficiency of the NLU component. Hence it could be better to combine the segmentation and NLU components in to a single end to end component.

We plan to extend our work to overcome the above mentioned limitations.

References

[1] Jianfeng Gao, Michel Galley, and Lihong Li. Neural approaches to conversational ai. *The 41st International ACM SIGIR Conference on Research Development in Information Retrieval - SIGIR 18*, 2018.

[2] Xiujun Li, Yun-Nung Chen, Lihong Li, Jianfeng Gao, and Asli Celikyilmaz. End-to-end task-completion neural dialogue systems, 2017, 1703.01008.

[3] Kevin Gurney. *An introduction to neural networks*. CRC press, 2014.

[4] Vlado Keselj. Speech and language processing daniel jurafsky and james h. martin (stanford university and university of colorado at boulder) pearson prentice hall, 2009, xxxi+ 988 pp; hardbound, isbn 978-0-13-187321-6, $115.00, 2009.

[5] Peter F. Brown, Peter V. deSouza, Robert L. Mercer, Vincent J. Della Pietra, and Jenifer C. Lai. Class-based n-gram models of natural language. *Comput. Linguist.*, 18(4):467–479, December 1992.

[6] Shane Bergsma, Dekang Lin, and Randy Goebel. Web-scale n-gram models for lexical disambiguation. In *Twenty-First International Joint Conference on Artificial Intelligence*, 2009.

[7] Nal Kalchbrenner and Phil Blunsom. Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1700–1709, 2013.

[8] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *CoRR*, abs/1607.04606, 2016, 1607.04606.

[9] Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. Learning sentiment-specific word embedding for twitter sentiment classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1555–1565, 2014.

[10] Prajit Ramachandran, Barret Zoph, and Quoc V Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[12] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, pages 807–814, USA, 2010. Omnipress.

[13] Andrew L. Maas. Rectifier nonlinearities improve neural network acoustic models. 2013.

[14] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In Yves Lechevallier and Gilbert Saporta, editors, *Proceedings of COMPSTAT'2010*, pages 177–186, Heidelberg, 2010. Physica-Verlag HD.

[15] Sebastian Ruder. An overview of gradient descent optimization algorithms, 2016, 1609.04747.

[16] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014, 1412.6980.

[17] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *NIPS*, 2017.

[18] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[19] Tomas Mikolov, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.

[20] Eric H. Huang, Richard Socher, Christopher D. Manning, and Andrew Y. Ng. Improving Word Representations via Global Context and Multiple Word Prototypes. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2012.

[21] Yun-Nung (Vivian) Chen, Dilek Hakkani-Tur, Gokhan Tur, Asli Celikyilmaz, Jianfeng Gao, and Li Deng. Syntax or semantics? knowledge-guided joint semantic frame parsing. December 2016.

[22] R. Sarikaya, G. E. Hinton, and A. Deoras. Application of deep belief networks for natural language understanding. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 22(4):778–784, April 2014.

[23] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017, 1706.03762.

[24] Bing Liu and Ian Lane. Attention-based recurrent neural network models for joint intent detection and slot filling. *Interspeech 2016*, Sep 2016.

[25] Puyang Xu and Ruhi Sarikaya. Exploiting shared information for multi-intent natural language sentence classification. August 2013.

[26] Byeongchang Kim, Seonghan Ryu, and Gary Geunbae Lee. Two-stage multi-intent detection for spoken language understanding. *Multimedia Tools and Applications*, 76(9):11377–11390, May 2017.

[27] Robert Dale, Hermann Moisl, and Harold Somers. *Handbook of natural language processing*. CRC Press, 2000.

[28] Rodrigo Agerri, Josu Bermudez, and German Rigau. Ixa pipeline: Efficient and ready to use multilingual nlp tools.

[29] Michael Elhadad. Natural language processing with python steven bird, ewan klein, and edward loper (university of melbourne, university of edinburgh, and bbn

technologies) sebastopol, ca: O'reilly media, 2009, xx+482 pp; paperbound, isbn 978-0-596-51649-9, $44.99; on-line free of charge at nltk.org/book. *Computational Linguistics*, 36:767–771, 2010.

[30] Arian L. Albert Genevieve B. Melton Serguei V.S. Pakhomov Benjamin C. Knoll, Elizabeth A. Lindemann. Recurrent deep network models for clinical nlp tasks: Use case with sentence boundary disambiguation. In *Proceedings of the 17th World Congress of Medical and Health Informatics*, MedInfo'19, USA, 2090.

[31] Yoshihiko Gotoh and Steve Renals. Sentence boundary detection in broadcast speech transcripts. In *ASR2000-Automatic Speech Recognition: Challenges for the new Millenium ISCA Tutorial and Research Workshop (ITRW)*, 2000.

[32] Yang Liu, Nitesh V Chawla, Mary P Harper, Elizabeth Shriberg, and Andreas Stolcke. A study in machine learning from imbalanced data for sentence boundary detection in speech. *Computer Speech & Language*, 20(4):468–494, 2006.

[33] Abderrahim Ait Azzi, Houda Bouamor, and Sira Ferradans. The finsbd-2019 shared task: Sentence boundary detection in pdf noisy text in the financial domain. In *The First Workshop on Financial Technology and Natural Language Processing in conjunction with IJCAI 2019*, page 74, 2019.

[34] George Sanchez. Sentence boundary detection in legal text. In *Proceedings of the Natural Legal Language Processing Workshop 2019*, pages 31–38, 2019.

[35] Chenglin Xu, Lei Xie, Guangpu Huang, Xiong Xiao, Eng Siong Chng, and Haizhou Li. A deep neural network approach for sentence boundary detection in broadcast news. In *Fifteenth annual conference of the international speech communication association*, 2014.

[36] Takao Doi and Eiichiro Sumita. Splitting input sentence for machine translation using language model with sentence similarity. In *Proceedings of the 20th International Conference on Computational Linguistics*, COLING '04, Stroudsburg, PA, USA, 2004. Association for Computational Linguistics.

[37] Erik F. Tjong, Kim Sang, and Hervé Déjean. Introduction to the conll-2001 shared task: Clause identification. In *Proceedings of the 2001 Workshop on Computational*

Natural Language Learning - Volume 7, ConLL '01, Stroudsburg, PA, USA, 2001. Association for Computational Linguistics.

[38] Osamu Furuse, Setsuo Yamada, and Kazuhide Yamamoto. Splitting long or ill-formed input for robust spoken-language translation. In *36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Volume 1*, 1998.

[39] Xiujun Li, Sarah Panda, Jingjing Liu, and Jianfeng Gao. Microsoft dialogue challenge: Building end-to-end task-completion dialogue systems. *arXiv preprint arXiv:1807.11125*, 2018.

Appendix A

Glossary and Acronyms

Care has been taken in this thesis to minimize the use of jargon and acronyms, but this cannot always be achieved. This section contains a table of acronyms and their meanings.

A.1 Acronyms

Table A.1: Acronyms

Acronym	Meaning
NLU	Natural Language Understanding
BOS	Beginning of Sentence
CNN	Convolutional Neural Networks
NLU	Natural Language Understanding
BiLSTM	Bidirectional Long Short Term Memory Networks
DST	Dialog State Tracker
ANN	Artificial Neural Network